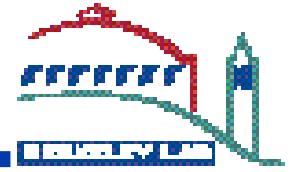


Web Application Development with JSP

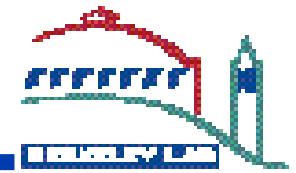
Greg Haverkamp
ITSD UNIX Systems Group
April 28, 2004

Overview



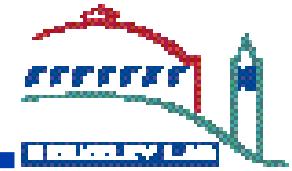
- **Overview of Java technologies**
- **Brief introduction to JSP with examples**
- **Database connectivity with JDBC**
- **Forms and JSP**
- **Other JSP/Servlet features**
- **Very quick Intro to Web application security**

Overview of Java Technologies



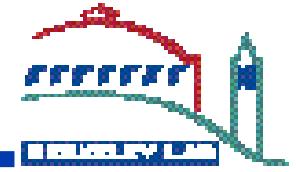
- **Java Virtual Machine**
- **Servlet API**
- **Servlet Container/JSP**
- **J2EE**

Java Virtual Machine



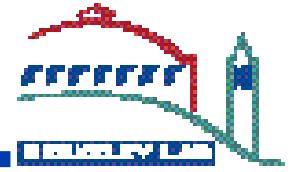
- Operating environment for Java applications
- Component that allows for the “write once, run everywhere” ability of Java
- Server-side Java is the one area where Java approaches this ideal

J2EE



- Java2 Enterprise Edition
- Describes JavaSoft's set of technologies for enterprise application development
- Includes Servlet API, JSP, JDBC, EJB, JavaMail, JMS, etc.

Servlet API



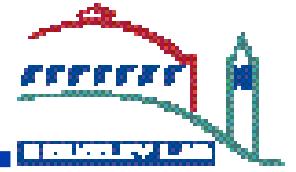
- Generalized API for the extension of servers with applications called Servlets
- Little used outside of HTTP
- HTTP subclass is HttpServlet

Servlet (J2EE) Container



- **Various available**
- **Free: Tomcat, JBOSS**
- **Inexpensive: Resin, JRun**
- **Expensive: WebLogic, WebSphere, iAS**

JSP



- **Java Servlet (HttpServlet in particular, in all common implementation)**
- **Servlet container compiles (or precompiles) at runtime**

Why JSP?



- **JSP and Servlet technologies are two of the most commonly seen in industry**
- **JSP and Servlet technologies are commonly used by enterprise application vendors**
- **Many excellent implementations from various vendors exist**
- **If you've got an existing investment in Java applications, JSP can easily bring those investments to the web**

JSP functionality



- Declaration: <%! ... %>

JSP functionality



- **Declaration:** `<%! ... %>`
- **Page directive:** `<%@ page ... %>`

JSP functionality



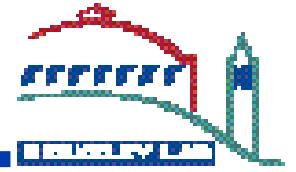
- **Declaration:** `<%! ... %>`
- **Page directive:** `<%@ page ... %>`
- **Expression:** `<%= ... %>`

JSP functionality



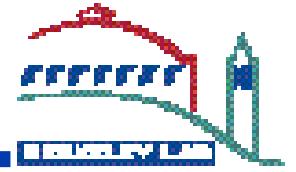
- **Declaration:** `<%! ... %>`
- **Page directive:** `<%@ page ... %>`
- **Expression:** `<%= ... %>`
- **Scriptlet:** `<% ... %>`

Simple example



- Basic HTML output from JSP
- Simple dynamic output from JSP

Basic HTML as JSP



```
<html>
<body>
Hello, world!
</body>
</html>
```

Basic dynamic JSP



```
<%@ page import="java.util.Calendar" %>
<%! int time =
    Calendar.getInstance().get(Calendar.AM_PM); %>
<html>
<body>
Good
<% if (time == Calendar.AM) { %>
Morning
<% } else { %>
Afternoon
<% } %>
, World!
</body>
</html>
```

Basic dynamic JSP



```
<%@ page import="java.util.Calendar" %>
<%! int time = Calendar.getInstance().get(Calendar.AM_PM); %>
<html>
<body>
Good
<% if (time == Calendar.AM) { %>
Morning
<% } else { %>
Afternoon
<% } %>
, World!
</body>
</html>
```

Basic dynamic JSP



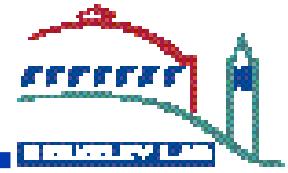
```
<%@ page import="java.util.Calendar" %>
<%! int time = Calendar.getInstance().get(Calendar.AM_PM); %>
<html>
<body>
Good
<% if (time == Calendar.AM) { %>
Morning
<% } else { %>
Afternoon
<% } %>
, World!
</body>
</html>
```

Basic dynamic JSP



```
<%@ page import="java.util.Calendar" %>
<%! int time = Calendar.getInstance().get(Calendar.AM_PM); %>
<html>
<body>
Good
<% if (time == Calendar.AM) { %>
Morning
<% } else { %>
Afternoon
<% } %>
, World!
</body>
</html>
```

JDBC



- **Unofficially, “Java Database Connectivity”**
- **Provides an abstract interface to various databases**
- **Similar to ODBC (Windows), DBI (Perl)**
- **Access mechanism remains the same. Database specifics (SQL syntax, connection strings, supported datatypes) still vary by database**

Using JDBC



- **Statement** - ad hoc queries
- **PreparedStatement** - pre-compiled queries
- **CallableStatement** - stored procedures (where supported)
- **ResultSet** - set of results returned from database

JDBC Example



- Simple table:

Field	Type	Null	Key	Default	Extra
<code>id</code>	<code>int(11)</code>		PRI	<code>NULL</code>	<code>auto_increment</code>
<code>makes</code>	<code>varchar(100)</code>	YES		<code>NULL</code>	

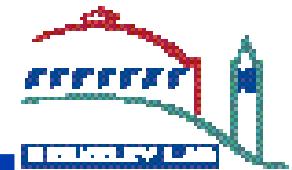
- Fill with values:

```
insert into makes values (null, 'BMW');
```

...

- Query with JDBC

JDBC Example Code



```
<%@ page import="javax.naming.*" %>
<%@ page import="javax.sql.*" %>
<%@ page import="java.sql.*" %>
<html>
<body>
<h2>Motorcycle Manufacturers</h2>
<p>
<ul><%
try{
    Context ctx = new InitialContext();
    if(ctx == null )
        throw new Exception("No Context");
    DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/TestDB");
    if (ds != null) {
        Connection conn = ds.getConnection();
        if(conn != null)  {
            Statement stmt = conn.createStatement();
            ResultSet rst = stmt.executeQuery( "select id, makes from makes");
            while(rst.next()) { %>
<li><%=rst.getString(2)%>
<%
                }
                conn.close();
            }
        }
    } catch(Exception e) {
        e.printStackTrace();
    }
%>
</ul>
</body>
</html>
```

JDBC Example Code



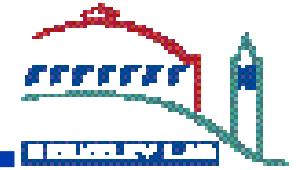
```
<%@ page import="javax.naming.*" %>
<%@ page import="javax.sql.*" %>
<%@ page import="java.sql.*" %>
<html>
<body>
<h2>Motorcycle Manufacturers</h2>
<p>
<ul><%
try{
    Context ctx = new InitialContext();
    if(ctx == null )
        throw new Exception("No Context");
    DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/TestDB");
    if (ds != null) {
        Connection conn = ds.getConnection();
        if(conn != null)  {
            Statement stmt = conn.createStatement();
            ResultSet rst = stmt.executeQuery( "select id, makes from makes");
            while(rst.next()) { %>
<li><%=rst.getString(2)%>
<%
                }
                conn.close();
            }
        }
    } catch(Exception e) {
        e.printStackTrace();
    }
%>
</ul>
</body>
</html>
```

JDBC Example Code



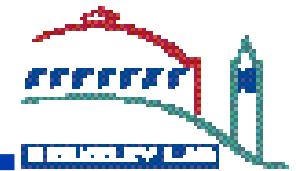
```
<%@ page import="javax.naming.*" %>
<%@ page import="javax.sql.*" %>
<%@ page import="java.sql.*" %>
<html>
<body>
<h2>Motorcycle Manufacturers</h2>
<p>
<ul><%
try{
    Context ctx = new InitialContext();
    if(ctx == null )
        throw new Exception("No Context");
    DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/TestDB");
    if (ds != null) {
        Connection conn = ds.getConnection();
        if(conn != null)  {
            Statement stmt = conn.createStatement();
            ResultSet rst = stmt.executeQuery( "select id, makes from makes");
            while(rst.next()) { %>
<li><%=rst.getString(2)%>
<%
                }
                conn.close();
            }
        }
    } catch(Exception e) {
        e.printStackTrace();
    }
%>
</ul>
</body>
</html>
```

JDBC Example Code



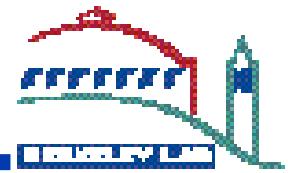
```
<%@ page import="javax.naming.*" %>
<%@ page import="javax.sql.*" %>
<%@ page import="java.sql.*" %>
<html>
<body>
<h2>Motorcycle Manufacturers</h2>
<p>
<ul><%
try{
    Context ctx = new InitialContext();
    if(ctx == null )
        throw new Exception("No Context");
    DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/TestDB");
    if (ds != null) {
        Connection conn = ds.getConnection();
        if(conn != null)  {
            Statement stmt = conn.createStatement();
            ResultSet rst = stmt.executeQuery( "select id, makes from makes");
            while(rst.next()) { %>
<li><%=rst.getString(2)%>
<%
                }
                conn.close();
            }
        }
    } catch(Exception e) {
        e.printStackTrace();
    }
%>
</ul>
</body>
</html>
```

JDBC Example Code



```
<%@ page import="javax.naming.*" %>
<%@ page import="javax.sql.*" %>
<%@ page import="java.sql.*" %>
<html>
<body>
<h2>Motorcycle Manufacturers</h2>
<p>
<ul><%
try{
    Context ctx = new InitialContext();
    if(ctx == null )
        throw new Exception("No Context");
    DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/TestDB");
    if (ds != null) {
        Connection conn = ds.getConnection();
        if(conn != null)  {
            Statement stmt = conn.createStatement();
            ResultSet rst = stmt.executeQuery( "select id, makes from makes");
            while(rst.next()) { %>
<li><%=rst.getString(2)%>
<%
                }
                conn.close();
            }
        }
    } catch(Exception e) {
        e.printStackTrace();
    }
%>
</ul>
</body>
</html>
```

Forms



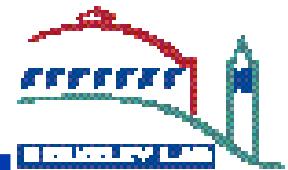
- **<form method="("post|get)" action="<target">">**
- **Form fields, such as text input fields, checkboxes, radio buttons, textareas, and buttons.**
- **In JSP, processing of fields handled in pre-defined *request* object.**

```
<form method="post" action="/DBTest/makes2.jsp">
  <input type=text name="max_makes" maxlength=10>
  <input type=submit>
```

- **In makes2.jsp, you would get “max_makes” like this:**

```
String maxMakesToShow = request.getParameter("max_makes");
```

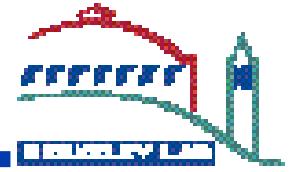
Forms - Insert and Update



- Use JDBC `executeUpdate`
- Should use caution when pulling from the environment

```
if(conn != null)  {  
    Statement stmt = conn.createStatement();  
    stmt.executeUpdate( "insert into makes values(null,  
'"+request.getParameter("new_make") +"' , " +  
request.getParameter("rating") + ")");  
}
```

Encapsulation



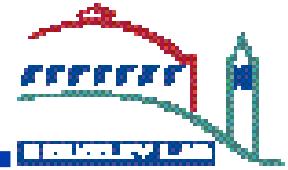
- We really want to keep code out of presentation as much as possible
- Key feature of tag-based languages -- no presentation in code
- Designers tend to dislike code; some design tools butcher code

How do we encapsulate code?



- In JSP:
 - Embed logic in objects
 - EJB
 - Straight JavaBeans
 - Tag libraries: even more designer-friendly

Tag libraries



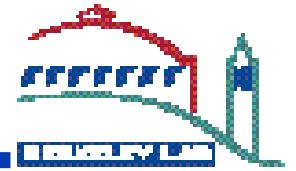
- Relatively new
- Encapsulates functionality
- Standard tags provided:
 - <jsp:useBean>
 - <jsp:setProperty>
 - <jsp:getProperty>
 - <jsp:include>
 - <jsp:forward>
- JSP Standard Tag Library (JSTL)
- Custom tags

Using standard tags



```
<%@ page import="foo.TaglibExample" %>
<html>
<body>
<h2>Motorcycle ManufacturerSelection</h2>
<p>
<jsp:useBean id="taglibExample" scope="page"
    class="foo.TaglibExample">
    <jsp:setProperty name="taglibExample" property="makeId"
        value="1"/>
</jsp:useBean>
The make you selected is <jsp:getProperty name="taglibExample"
    property="makeId"/>
</body>
</html>
```

Using standard tags



```
<%@ page import="foo.TaglibExample" %>
<html>
<body>
<h2>Motorcycle ManufacturerSelection</h2>
<p>
<jsp:useBean id="taglibExample" scope="page"
    class="foo.TaglibExample">
    <jsp:setProperty name="taglibExample" property="makeId"
        value="1"/>
</jsp:useBean>
The make you selected is <jsp:getProperty name="taglibExample"
    property="makeId"/>
</body>
</html>
```

Using standard tags



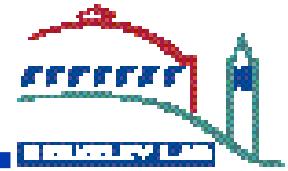
```
<%@ page import="foo.TaglibExample" %>
<html>
<body>
<h2>Motorcycle ManufacturerSelection</h2>
<p>
<jsp:useBean id="taglibExample" scope="page"
    class="foo.TaglibExample">
    <jsp:setProperty name="taglibExample" property="makeId"
        value="1"/>
</jsp:useBean>
The make you selected is <jsp:getProperty name="taglibExample"
    property="makeId"/>
</body>
</html>
```

Using standard tags



```
<%@ page import="foo.TaglibExample" %>
<html>
<body>
<h2>Motorcycle ManufacturerSelection</h2>
<p>
<jsp:useBean id="taglibExample" scope="page"
    class="foo.TaglibExample">
    <jsp:setProperty name="taglibExample" property="makeId"
        value="1"/>
</jsp:useBean>
The make you selected is <jsp:getProperty name="taglibExample"
    property="makeId"/>
</body>
</html>
```

Using standard tags



```
<%@ page import="foo.TaglibExample" %>
<html>
<body>
<h2>Motorcycle Manufacturers</h2>
<p>
<jsp:useBean id="taglibExample" scope="page" class="foo.TaglibExample">
    <jsp:setProperty name="taglibExample" property="*" />
</jsp:useBean>
```

The make you selected is <jsp:getProperty name="taglibExample" property="makeId"/>

```
<p>
Make number to display:
<form method="get" action="taglib2.jsp">
<input type="text" name="makeId">
<input type="submit">
</form>

</body>
</html>
```

Custom tags



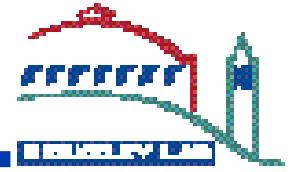
- Allows functionality, logic in tags
- Insulates page developers from logic and scriptlets

Our first dynamic JSP



```
<%@ page import="java.util.Calendar" %>
<%! int time =
    Calendar.getInstance().get(Calendar.AM_PM); %>
<html>
<body>
Good
<% if (time == Calendar.AM) { %>
Morning
<% } else { %>
Afternoon
<% } %>
, World!
</body>
</html>
```

Custom tag example



```
<%@ taglib uri="/WEB-INF/jsp/mytaglib.tld" prefix="mytags" %>
<html>
<body>
Good <mytags:ampm/>, World!
</body>
</html>
```

Custom tag example



```
public int doStartTag() throws JspException {
    int time = Calendar.getInstance().get(Calendar.AM_PM);
    if (time == Calendar.AM) {
        this.AM_PM = this.AM;
    } else {
        this.AM_PM = this.PM;
    }
    try {
        pageContext.getOut().print(this.AM_PM);
    } catch (IOException ie) {
        throw new JspException("Error while writing to
client.");
    }

    return SKIP_BODY;
}
```

Web Application Security



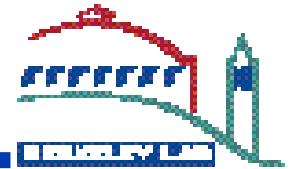
- **Most of our examples somewhat contrived; no significant diligence put into security**
- **When applications will be opened up to the world, security is important**

Why do I care about security?



- Java lulls developers into a false sense of security
- Sandbox adds security
- Root may not be achieved in correctly implemented systems
- Data is still at risk!

Short security example



- Let's say I rewrite our earlier example4:

```
if(conn != null)  {
    Statement stmt = conn.createStatement();
    ResultSet rst = stmt.executeQuery( "select id, make from
makes where rating = 1");
    while(rst.next() && (maxMakesToShow-- != 0)) { %>
<li><%=rst.getString(2)%>
<%
}
}

if(conn != null)  {
    Statement stmt = conn.createStatement();
    ResultSet rst = stmt.executeQuery(
        "select id, make from makes where (rating <= 1) and (id <= " +
maxId + ")");
    while(rst.next()) { %>
<li><%=rst.getString(2)%>
<%
}
}
```

Short security example



- URL mangling and ad hoc queries can produce bad results:

```
http://localhost:8080/jsp-examples/example9.jsp?maxId=3)%20or%20(1=1
```

Conclusion



- **JSP and Servlets are powerful, cross-platform enterprise development components for the Java platform**
- **JSP provides great flexibility and development speed**
- **JSP provides considerable amounts of customization capability**
- **Care must be exercised whenever applications will find themselves on the Internet**